
Subject: [PATCH 1/2] Introduce iplink_parse() routine
Posted by [Pavel Emelianov](#) on Thu, 19 Jul 2007 09:32:31 GMT
[View Forum Message](#) <> [Reply to Message](#)

This routine parses CLI attributes, describing generic link parameters such as name, address, etc.

This is mostly copy-pasted from iplink_modify().

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

Acked-by: Patrick McHardy <kaber@trash.net>

```
include/utils.h | 3 +
ip/iplink.c      | 127 ++++++
2 files changed, 76 insertions(+), 54 deletions(-)
```

```
diff --git a/include/utils.h b/include/utils.h
```

```
index a3fd335..3fd851d 100644
```

```
--- a/include/utils.h
```

```
+++ b/include/utils.h
```

```
@@ -146,4 +146,7 @@ extern int cmdlineno;
extern size_t getcmdline(char **line, size_t *len, FILE *in);
extern int makeargs(char *line, char *argv[], int maxargs);
```

```
+struct iplink_req;
```

```
+int iplink_parse(int argc, char **argv, struct iplink_req *req,
```

```
+ char **name, char **type, char **link, char **dev);
```

```
#endif /* __UTILS_H__ */
```

```
diff --git a/ip/iplink.c b/ip/iplink.c
```

```
index cfacdab..e0e0d85 100644
```

```
--- a/ip/iplink.c
```

```
+++ b/ip/iplink.c
```

```
@@ -142,140 +142,159 @@ static int iplink_have_newlink(void)
```

```
}
```

```
#endif /* ! IPLINK_IOCTL_COMPAT */
```

```
-static int iplink_modify(int cmd, unsigned int flags, int argc, char **argv)
```

```
+struct iplink_req {
```

```
+ struct nlmsg_hdr n;
```

```
+ struct ifinfomsg i;
```

```
+ char buf[1024];
```

```
+};
```

```
+
```

```
+int iplink_parse(int argc, char **argv, struct iplink_req *req,
```

```
+ char **name, char **type, char **link, char **dev)
```

```
{
```

```

+ int ret, len;
+ char abuf[32];
  int qlen = -1;
  int mtu = -1;
- int len;
- char abuf[32];
- char *dev = NULL;
- char *name = NULL;
- char *link = NULL;
- char *type = NULL;
- struct link_util *lu = NULL;
- struct {
-   struct nlmsgghdr n;
-   struct ifinfomsg i;
-   char  buf[1024];
- } req;

- memset(&req, 0, sizeof(req));
-
- req.n.nlmsg_len = NLMSG_LENGTH(sizeof(struct ifinfomsg));
- req.n.nlmsg_flags = NLM_F_REQUEST|flags;
- req.n.nlmsg_type = cmd;
- req.i.ifl_family = preferred_family;
+ ret = argc;

  while (argc > 0) {
    if (strcmp(*argv, "up") == 0) {
-   req.i.ifl_change |= IFF_UP;
-   req.i.ifl_flags |= IFF_UP;
+   req->i.ifl_change |= IFF_UP;
+   req->i.ifl_flags |= IFF_UP;
    } else if (strcmp(*argv, "down") == 0) {
-   req.i.ifl_change |= IFF_UP;
-   req.i.ifl_flags &= ~IFF_UP;
+   req->i.ifl_change |= IFF_UP;
+   req->i.ifl_flags &= ~IFF_UP;
    } else if (strcmp(*argv, "name") == 0) {
      NEXT_ARG();
-   name = *argv;
+   *name = *argv;
    } else if (matches(*argv, "link") == 0) {
      NEXT_ARG();
-   link = *argv;
+   *link = *argv;
    } else if (matches(*argv, "address") == 0) {
      NEXT_ARG();
      len = ll_addr_a2n(abuf, sizeof(abuf), *argv);
-   addattr_l(&req.n, sizeof(req), IFLA_ADDRESS, abuf, len);

```

```

+  addattr_l(&req->n, sizeof(*req), IFLA_ADDRESS, abuf, len);
} else if (matches(*argv, "broadcast") == 0 ||
-  strcmp(*argv, "brd") == 0) {
+  strcmp(*argv, "brd") == 0) {
    NEXT_ARG();
    len = ll_addr_a2n(abuf, sizeof(abuf), *argv);
-  addattr_l(&req.n, sizeof(req), IFLA_BROADCAST, abuf, len);
+  addattr_l(&req->n, sizeof(*req), IFLA_BROADCAST, abuf, len);
} else if (matches(*argv, "txqueuelen") == 0 ||
-  strcmp(*argv, "qlen") == 0 ||
-  matches(*argv, "txqlen") == 0) {
+  strcmp(*argv, "qlen") == 0 ||
+  matches(*argv, "txqlen") == 0) {
    NEXT_ARG();
    if (qlen != -1)
        duparg("txqueuelen", *argv);
    if (get_integer(&qlen, *argv, 0))
        invarg("Invalid \"txqueuelen\" value\n", *argv);
-  addattr_l(&req.n, sizeof(req), IFLA_TXQLEN, &qlen, 4);
+  addattr_l(&req->n, sizeof(*req), IFLA_TXQLEN, &qlen, 4);
} else if (strcmp(*argv, "mtu") == 0) {
    NEXT_ARG();
    if (mtu != -1)
        duparg("mtu", *argv);
    if (get_integer(&mtu, *argv, 0))
        invarg("Invalid \"mtu\" value\n", *argv);
-  addattr_l(&req.n, sizeof(req), IFLA_MTU, &mtu, 4);
+  addattr_l(&req->n, sizeof(*req), IFLA_MTU, &mtu, 4);
} else if (strcmp(*argv, "multicast") == 0) {
    NEXT_ARG();
-  req.i.ifi_change |= IFF_MULTICAST;
+  req->i.ifi_change |= IFF_MULTICAST;
    if (strcmp(*argv, "on") == 0) {
-  req.i.ifi_flags |= IFF_MULTICAST;
+  req->i.ifi_flags |= IFF_MULTICAST;
    } else if (strcmp(*argv, "off") == 0) {
-  req.i.ifi_flags &= ~IFF_MULTICAST;
+  req->i.ifi_flags &= ~IFF_MULTICAST;
    } else
        return on_off("multicast");
} else if (strcmp(*argv, "allmulticast") == 0) {
    NEXT_ARG();
-  req.i.ifi_change |= IFF_ALLMULTI;
+  req->i.ifi_change |= IFF_ALLMULTI;
    if (strcmp(*argv, "on") == 0) {
-  req.i.ifi_flags |= IFF_ALLMULTI;
+  req->i.ifi_flags |= IFF_ALLMULTI;
    } else if (strcmp(*argv, "off") == 0) {

```

```

- req.i.ifi_flags &= ~IFF_ALLMULTI;
+ req->i.ifi_flags &= ~IFF_ALLMULTI;
} else
    return on_off("allmulticast");
} else if (strcmp(*argv, "promisc") == 0) {
    NEXT_ARG();
- req.i.ifi_change |= IFF_PROMISC;
+ req->i.ifi_change |= IFF_PROMISC;
    if (strcmp(*argv, "on") == 0) {
- req.i.ifi_flags |= IFF_PROMISC;
+ req->i.ifi_flags |= IFF_PROMISC;
    } else if (strcmp(*argv, "off") == 0) {
- req.i.ifi_flags &= ~IFF_PROMISC;
+ req->i.ifi_flags &= ~IFF_PROMISC;
    } else
        return on_off("promisc");
    } else if (strcmp(*argv, "trailers") == 0) {
        NEXT_ARG();
- req.i.ifi_change |= IFF_NOTRAILERS;
+ req->i.ifi_change |= IFF_NOTRAILERS;
        if (strcmp(*argv, "off") == 0) {
- req.i.ifi_flags |= IFF_NOTRAILERS;
+ req->i.ifi_flags |= IFF_NOTRAILERS;
        } else if (strcmp(*argv, "on") == 0) {
- req.i.ifi_flags &= ~IFF_NOTRAILERS;
+ req->i.ifi_flags &= ~IFF_NOTRAILERS;
        } else
            return on_off("trailers");
    } else if (strcmp(*argv, "arp") == 0) {
        NEXT_ARG();
- req.i.ifi_change |= IFF_NOARP;
+ req->i.ifi_change |= IFF_NOARP;
        if (strcmp(*argv, "on") == 0) {
- req.i.ifi_flags &= ~IFF_NOARP;
+ req->i.ifi_flags &= ~IFF_NOARP;
        } else if (strcmp(*argv, "off") == 0) {
- req.i.ifi_flags |= IFF_NOARP;
+ req->i.ifi_flags |= IFF_NOARP;
        } else
            return on_off("noarp");
#ifdef IFF_DYNAMIC
    } else if (matches(*argv, "dynamic") == 0) {
        NEXT_ARG();
- req.i.ifi_change |= IFF_DYNAMIC;
+ req->i.ifi_change |= IFF_DYNAMIC;
        if (strcmp(*argv, "on") == 0) {
- req.i.ifi_flags |= IFF_DYNAMIC;
+ req->i.ifi_flags |= IFF_DYNAMIC;

```

```

    } else if (strcmp(*argv, "off") == 0) {
-   req.i.ifi_flags &= ~IFF_DYNAMIC;
+   req->i.ifi_flags &= ~IFF_DYNAMIC;
    } else
        return on_off("dynamic");
#endif
    } else if (matches(*argv, "type") == 0) {
        NEXT_ARG();
-   type = *argv;
+   *type = *argv;
        argc--; argv++;
        break;
    } else {
-       if (strcmp(*argv, "dev") == 0) {
+   if (strcmp(*argv, "dev") == 0) {
        NEXT_ARG();
        }
-   if (dev)
+   if (*dev)
        duparg2("dev", *argv);
-   dev = *argv;
+   *dev = *argv;
        }
        argc--; argv++;
    }

+ return ret - argc;
+}
+
+static int iplink_modify(int cmd, unsigned int flags, int argc, char **argv)
+{
+ int len;
+ char *dev = NULL;
+ char *name = NULL;
+ char *link = NULL;
+ char *type = NULL;
+ struct link_util *lu = NULL;
+ struct iplink_req req;
+ int ret;
+
+ memset(&req, 0, sizeof(req));
+
+ req.n.nlmsg_len = NLMSG_LENGTH(sizeof(struct ifinfomsg));
+ req.n.nlmsg_flags = NLM_F_REQUEST|flags;
+ req.n.nlmsg_type = cmd;
+ req.i.ifi_family = preferred_family;
+
+ ret = iplink_parse(argc, argv, &req, &name, &type, &link, &dev);

```

```
+ if (ret < 0)
+ return ret;
+
+ argc -= ret;
+ argv += ret;
  ll_init_map(&rth);

  if (type) {
```
